UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Robert Vurušič

Vgrajen sistem za avtomobilsko diagnostiko z vmesnikom Bluetooth

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Robert Vurušič

Vgrajen sistem za avtomobilsko diagnostiko z vmesnikom Bluetooth

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: izr. prof. dr. Patricio Bulić

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Načrtujte vgrajen sistem z vmesnikom Bluetooth za avtomobilsko diagnostiko. Vgrajen sistem naj uporablja standardni priključek z vmesnikom Bluetooth v avtomobilih. Za implementacijo sistema uporabite razvojno ploščico STM32F Discovery z mikrokrmilnikom ARM Cortex-M4. Sistem naj podatke pošilja na mobilno napravo z operacijskim sistemom Android preko vmesnika Bluetooth.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Robert Vurušič, z vpisno številko **63980178**, sem avtor diplomskega dela z naslovom:

Prikazovalnik stanja motorja.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. septembra 2014 Podpis avtorja:

Zahvalil bi se svoji ženi Petri in vsem trem otrokom za spodbudo in njim ukraden čas. Zahvala gre mojemu mentorju in profesorju izr. prof. dr. Patriciu Buliću, ker s svojim delom in odnosom dokazuje, da je še upanje za naše šolstvo. Zahvaljujem se prijatelju Matevžu Pesku za pomoč pri prvih korakih v svet Jave, Marinki Miklavčič za lektoriranje, itd.

Mojim malim pingvinom, ki so moja prihodnost

Kazalo

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
BT	Bluetooth	Modri zob
CAN	Controller Area Network	Krmilniško omrežje
DB9	D-Sub type of connector	Priključek tipa D-sub
DTC	Diagnostic Trouble Code	Diagnostična koda napake
GPIO	General Purpose Input/Output	Splošnonamenski vhod/izhod
ISM	Industrial, scientific and medical (ISM)	Industrijski, znanstveni, medicinski
OBD	On Board Diagnostics	Avtomobilska diagnostika
PID	Parameter ID	Identifikacijski parameter
UART	Universal asynchronous receiver/transmitter	Univerzalen asinhronski sprejemnik in oddajnik
USART	Universal synchronous asynchronous receiver/transmitter	Univerzalen sinhronski asinhronski sprejemnik in oddajnik
USB	Universal Serial Bus	Univerzalno zaporedno vodilo

Povzetek

V avtomobilski industriji se uporablja diagnostika OBDII (On Board Diagnostics II) na osnovi vodila CAN (Controller Area Network) za spremljanje stanja vozila. Elektronski sistemi v vozilu so povezani v omrežje, nanj pa se lahko poveže tudi zunanji uporabnik in spremlja celoten promet na omrežju ali pa poizveduje le po določenih vrednostih. Tako lahko pridobi podatke, kot so hitrost ali temperatura hladilne tekočine, lahko pa pridobi tudi podatke o napakah ali nepravilnem delovanju sistema. V diplomski nalogi smo želeli sestaviti sistem na podlagi mikrokrmilnika STM32F407, ki najprej posreduje podatke z vodila CAN preko žične povezave mikrokrmilniku. Nato pa te podatke brezžično posreduje na napravo z operacijskim sistemom Android. Na napravi Android se podatki osvežujejo in prikažejo v uporabniku prijazni obliki. Potrebno je bilo povezati vodilo CAN, mikrokrmilnik, povezavo Bluetooth in aplikacijo za Android. Ker je mikrokrmilnik zmogljiva naprava in naprava Android priklopljena na informacijsko omrežje, so možnosti nadgradnje široke.

Ključne besede: diagnostika, CAN, OBDII, mikrokrmilnik, ST32F407, Bluetooth, Android

Abstract

In the automotive industry, the OBDII (On Board Diagnostics II) diagnostics based on CAN (Controller Area Network) bus is used for monitoring the condition of the vehicle. Electronic systems in the vehicle are connected to a network. Outside user can to this network and monitor the entire traffic on the network or inquires only certain values. You can obtain information such as speed or coolant temperature, but you can also obtain information about the error or malfunction of the system. In our thesis we want to assemble a system based on microcontroller STM32F407 Discovery, which collects data from CAN bus and transmits via wired connection to microcontroller. Then this data are wirelessly transmited to Android device. On the Android device the data are periodically refreshed and displayed in user-friendly form. It was necessary to connect the CAN bus, microcontroller, Bluetooth connection and application for Android. Upgrade options are good, since the microcontroller is powerful device and Android device is connected to the information network.

Keywords: diagnostics, CAN, OBDII, microcontroler, ST32F407, Bluetooth, Android

Poglavje 1 Uvod

Začetki avtomobilske diagnostike segajo v 80. leta 20. stoletja, ko so posamezni proizvajalci začeli povezovati elektronske sisteme v avtomobilih za različne potrebe. Razvoj je bil pospešen v 90. letih, ko so se pojavile zahteve po spremljanju parametrov avtomobila, ki vplivajo na onesnaževanje okolja. Razvit je bil standard OBDII, ki je po letu 2004 obvezen za vsa proizvedena vozila. Po tem standardu lahko s pomočjo posebnega priključka v potniški kabini vozila dostopamo do parametrov vozila. Pridobiti je možno vse parametre, ki se prikazujejo na armaturni plošči vozila pa tudi mnoge druge ter tudi podrobnejše informacije o napakah na vozilu. Na trgu obstaja mnogo rešitev, ki rešujejo pridobivanje in prikaz informacij, v naši diplomski nalogi pa smo želeli to izvesti na osnovi mikrokrmilnika ST32F407, brezžično in s prikazom na napravi Android.

Najprej smo v diplomskem delu širše pogledali vodilo CAN, ki celotnemu sistemu omogoča delovanje. Opisali smo kratko zgodovino ter kako vodilo CAN deluje. Sledi opis standarda OBDII, ki izkorišča samo vodilo in naredi diagnostiko univerzalno za uporabo. Opisan pa je tudi uporabljeni sprejemno-oddajni modul CAN, ki je potreben za povezavo mikrokrmilnika z vodilom CAN. Ker smo želeli brezžično povezavo z napravo Android, smo opisali kratko zgodovino in delovanje povezave Bluetooth ter sam uporabljen modul Bluetooth HC-05. Kratko smo opisali tudi zgodovino operacijskega sistema Android in napravo, na kateri se izvaja to razvojno okolje. Predstavili smo tudi srce sistema, mikrokrmilnik ST32F407, ki vse povezuje, ter okolje EWARM, v katerem se razvijajo programi za sam mikrokrmilnik.

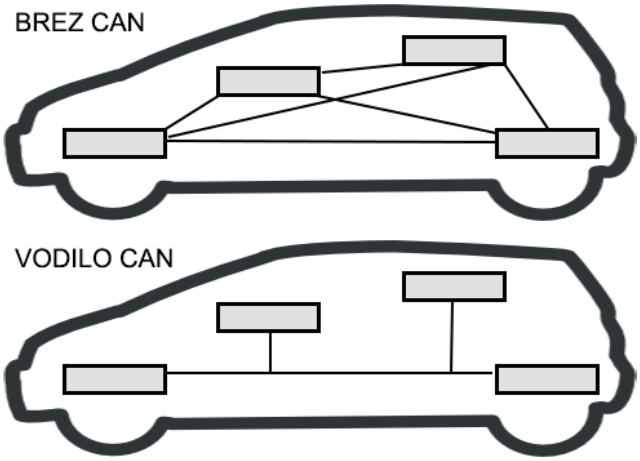
Sledi razlaga, kako smo sam sistem implementirali. Za vsako posamezno enoto je predstavljen strojni in programski del s primeri kode, ki smo jo uporabili. Na koncu smo opisali delovanje sistema in ugotovitve, do katerih smo prišli ob razvoju sistema.

Poglavje 2 Osnovni gradniki

2.1 Zajem podatkov z avtomobilskega vodila CAN

2.1.1 Kratka zgodovina

Omrežje CAN (Controller Area Network) je razvilo podjetje Bosch leta 1985 za povezavo elektronskih sistemov v vozilih. Predhodno se je uporabljal sistem »Point-to-point«, kar pomeni, da je bila povezava med vsako elektronsko napravo, kot je razvidno iz prvega dela slike 1.



Slika 1 : CAN »Point-to-point«

Ker so vozila vsebovala vedno več elektronskih sistemov, ki so morali biti povezani, je bil dotedanji pristop neprimeren. Vedno več vodnikov oz. povezav je posledično pomenilo:

- več elektronskih motenj
- večjo težo
- kompleksnost
- strošek pri proizvodnji

Uvedeno je bilo omrežje CAN, ki potrebuje občutno manj vodnikov, posledično pa je na omrežju manj motenj in je cenejše za proizvodnjo. Zaradi vseh teh prednosti je bilo omrežje CAN hitro sprejeto v avtomobilski industriji in že leta 1994 je bilo določeno kot standard ISO 11898. V naslednjih letih so se na podlagi tega standarda izpeljali še drugi, kot sta CANopen in DeviceNet, ki se uporabljata v industriji [1]. Kako je potekal razvoj, je razvidno iz tabele 1.

Leto	Dogodek
1983	Začetek projekta omrežja za avtomobile v podjetju Bosch
1986	Uradna predstavitev protokola CAN
1987	Prvi čipi kontrolerjev za CAN podjetij Intel in Philips Semiconductors
1991	Podjetje Bosch objavi specifikacije za CAN
1991	Predstavljen je bil CAN Kingdom, na protokolu CAN osnovan visokonivojski protokol – predstavilo podjetje Kvaser
1992	Ustanovljena skupina uporabnikov in proizvajalcev CAN in Automation (CiA)
1992	CiA objavila protokol za aplikacijsko plast CAN
1993	Objavljen standard ISO 11898
1994	CiA organizira prvo mednarodno konferenco CAN
1994	Allen-Bradley predstavi protokol DeviceNet
1995	Objavljen ISO 11898 sprememba – format razširjenega okvirja
1995	CiA objavila protokol CANopen
2000	Razvoj časovno prožnega CAN protokola TTCAN

Tabela 1: Potek zgodovinskega razvoja [2]

Omrežje CAN se danes uporablja v avtomatizaciji, železniških sistemih in podobnih sistemih. Osredotočili se bomo predvsem na ISO 11898-1, ki se ga uporablja za povezavo elektronskih sistemov v vozilih.

2.1.2 Delovanje protokola CAN

Standard 2.0 za CAN je razdeljen na dva dela:

- del A, ki določa osnovni format okvirja CAN (»base frame format«)
- del B, ki določa osnovni in razširjeni format okvirja CAN (»extended frame format«)

Razlika je v dolžini okvirja, kjer je osnovni format okvirja CAN dolg 11 bitov, razširjeni format okvirja CAN pa 29 bitov. Posledično ima razširjeni format nižjo odzivnost, zahteva več pasovne širine in ima slabšo detekcijo napak [3]. Pri protokolu CAN so od standardiziranih plasti (ISO model) predvsem prisotne povezovalna, fizična in aplikacijska plast.

Lastnosti protokola CAN:

- ker se pri protokolu CAN uporablja dominantne in recesivne bite, se lahko za prenosni medij izbere poljuben vodnik (bakrena žica, optični vodnik …)
- naprave nimajo svojega naslova – vsebino prepoznavajo s filtriranjem okvirjev, ki so vsebinsko določeni
- dodajanje novih članov omrežja je preprosto, teoretično neomejeno
- uporablja se kodiranje NRZ (»non-return-to-zero«)

Pri protokolu CAN so določeni štirje tipi okvirjev, s pomočjo katerih je definiran, in so navedeni v tabeli 2 [4].

Tip okvirja	Opis polj
Podatkovni	Namenjen je prenosu podatkov in vsebuje polja SoF (Start of Frame) (1 bit), arbitražno polje (12 bitov), kontrolno polje (6 bitov), podatkovno polje (do 64 bitov), polje CRC (16 bitov), polje za potrditev ACK (2 bita), zaključek okvirja (7 bitov)
Daljinski	Namenjen je zahtevi podatkov in vsebuje polja SoF (Start of Frame) (1 bit), arbitražno polje (12 bitov), kontrolno polje (6 bitov), polje CRC (16 bitov), polje za potrditev ACK (2 bita), zaključek okvirja (7 bitov)
Prekoračitveni	Okvir je namenjen obveščanju o preobremenitvi omrežja oz. člana omrežja
Okvir napake	Okvir se pošlje, ko se ugotovi, da je prišlo do napake v okvirju; rezultat je, da se okvir pošlje še enkrat

Tabela 2: Tipi okvirjev CAN

V avtomobilski industriji sta najbolj sprejeta standarda ISO 11898-2 High-Speed CAN (CAN visoke hitrosti) in ISO 11898-3 Fault Tolerant Low Speed CAN (CAN nizke hitrosti, odporen na napake) [5]. Primeri uporabe so v tabeli 3.

ISO 11898-2	ISO 11898-3
krmilnik ABS	krmilnik zračnih blazin
krmilnik servovolana	krmilnik svetil
krmilnik ESC	krmilnik klima naprave
uporablja se, kjer je potrebna visoka odzivnost	uporablja se za manj pomembne sisteme

Tabela 3: Primeri uporabe standardov

2.1.3 OBD-II

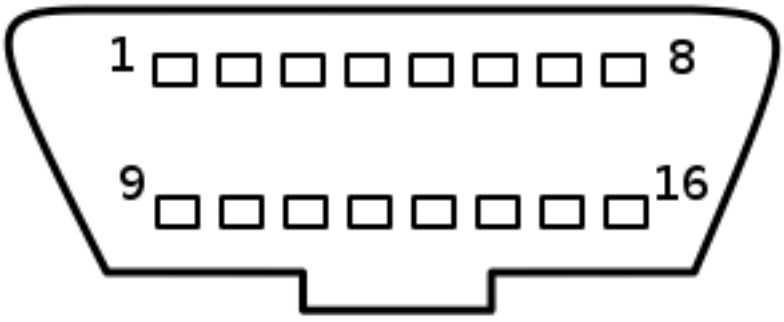
Protokol CAN je eden od protokolov, ki se uporabljajo pri avtomobilski diagnostiki OBD II (»on-board diagnostics«). Avtomobilska diagnostika OBD II se uporablja za spremljanje stanja avtomobila (trenutni parametri, napake …) in je obvezna za vsa vozila, proizvedena po letu 2004. Prvotno je bil zasnovan za spremljanje parametrov, ki pri avtomobilu onesnažujejo okolje, kasneje pa za spremljanje stanja ostalih elektronskih sistemov v avtomobilu.

Tehnično je izveden s priključkom, ki podpira več protokolov. Za naš primer bomo pogledali ISO 15765, ki uporablja protokol CAN.

Priključek ima 16 pinov in je trapezaste oblike. Za ISO 15765 so uporabljeni pini:

- pin 6 – CAN-High
- pin 16 – CAN-Low
- pin 5 – GND

Razporeditev pinov je razvidna na sliki 2.



Slika 2: OBD II priključek [6]

Znotraj standarda OBD II imamo na voljo 10 načinov delovanja. Vidni so v tabeli 4.

Koda	Opis načina delovanja
01	Trenutni podatki
02	Zamrznjeni podatki
03	Prikaži napake (DTC)
04	Počisti napake (DTC)
05	Test tipala za kisik (vozila brez CAN)
06	Test vseh tipal, tudi za kisik (vozila z CAN)
07	Trenutne napake (v zadnji vožnji)
08	Preverjanje vgrajenih komponent oz. sistema
09	Podatki o vozilu
0A	Stalne napake (DTC)

Tabela 4: Načini delovanja pri OBD II

OBD II deluje na podlagi pošiljanja (poizvedba) in prejemanja (odgovor) parametrov [7]. Za informacijo, ki jo želimo prejeti, moramo poslati okvir, sestavljen glede na zahteve:

- okvir mora imeti oznako med 0x7E0 in 0x7E7 – na te oznake se odzivajo krmilne enote; če želimo nasloviti vse naenkrat, pošljemo oznako 0x7DF
- v okvirju je nato vrednost 0x02, ki pomeni število dodatnih bajtov (1 bajt)
- določiti moramo način delovanja – 0x01 pomeni trenutne vrednosti (1 bajt)
- določiti moramo PID (Parameter ID), ki določa, katero vrednost želimo (1 bajt)
- okvir se zaključi s poljubnimi vrednostmi, saj se ne upoštevajo (5 bajtov)

Nekaj najbolj pogostih parametrov PID [8], ki jih lahko pridobimo v načinu 0x01, je navedenih v tabeli 5.

PID	Opis
0x04	obremenitev motorja
0x05	temperatura hladilne tekočine
0x0C	vrtljaji motorja
0x0D	hitrost vozila
0x0F	temperatura vstopnega zraka

Tabela 5: Primeri parametrov PID

Odgovor pa je sestavljen:

- na začetku okvirja je vrednost med 0x03 in 0x06, ki pomeni število dodatnih bajtov (1 bajt)
- sledi način delovanja – 0x01, ki pa ima odmik 0x40 (1 bajt)
- prejmemo PID podatke, ki smo jih poslali (1 bajt)
- nato prejmemo zahtevane podatke (4 bajte)
- okvir se zaključi s poljubno vrednostjo, saj se ne upoštevajo (1 bajt)

Vsak PID ima svojo enačbo, po kateri izračunamo njegovo točno vrednost. 4 bajte, ki jih prejmemo, lahko označimo kot A, B, C in D. Najbolj pogosto sta uporabljena le prva dva. Primer uporabe je v tabeli 6.

PID	Opis	Enačba za izračun
0x04	Obremenitev motorja	*A*100/255*
0x05	Temperatura hladilne tekočine	*A–40*
0x0C	Vrtljaji motorja	((A*256)+B/4
0X0D	Hitrost vozila	A

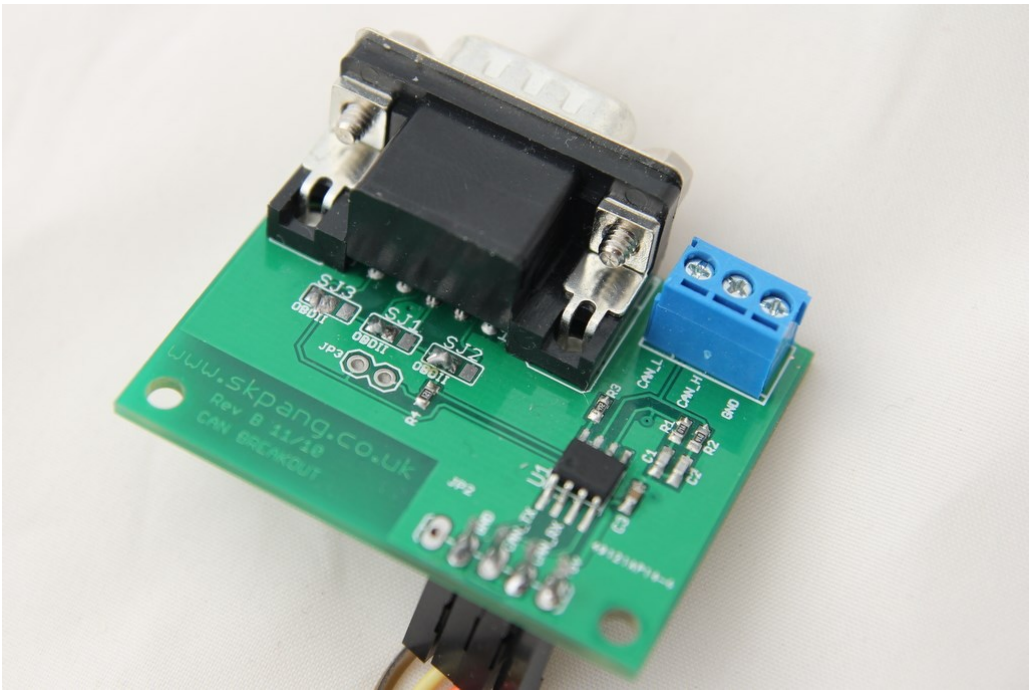
Tabela 6: Primer izračuna vrednosti PID

2.1.4 Modul CAN

Za naš diagnostični sistem smo potrebovali povezavo med protokolom CAN in mikrokrmilnikom ARM STM32F407. To smo rešili z razširitvenim modulom za vodilo CAN (»CAN-Bus Breakout Board«), ki ima lastnosti:

- sprejemnik/oddajnik CAN čip MCP2551
- napajanje 5 V
- podpira hitrost do 1 Mb/s
- implementiran ISO-11898 standard za fizični nivo
- do 112 priklopljenih enot
- priključek DB9

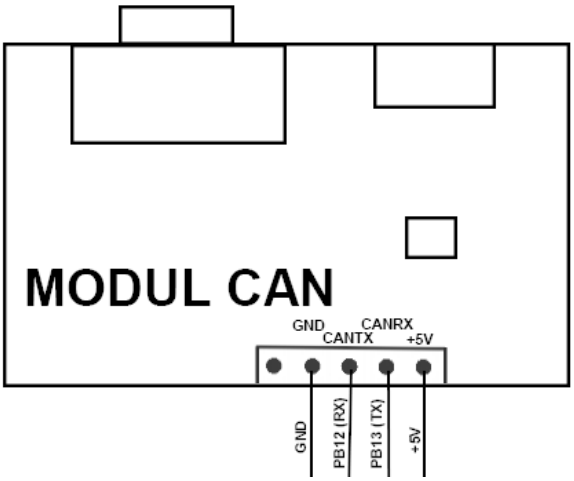
Na priključek DB9 razširitvenega modula CAN s slike 3 smo priklopili kabel s priključkom DB9 - OBD-II, ki je na sliki 4. Tako smo povezali modul z avtomobilom.



Slika 3: Razširitveni modul CAN

Slika 4: Kabel DB9 – OBD II

Z mikrokrmilnikom pa smo povezali modul preko povezave CAN. Za to povezavo je bilo potrebno privariti na modulu CAN 4-žilni kabel na 4 pine (GND, CAN_TX,CAN_RX, +5V), kot kaže slika 5. Predhodno pa smo še morali povezati ustrezne povezave SJ1, SJ2 in SJ3 na modulu, da smo nastavili ustrezno delovanje modula. Omogoča delovanje kot analizator CAN ali pa kot OBD II. V našem primeru smo potrebovali slednje (OBD II) in smo na vseh 3 povezavah povezali leva polja, da deluje v tem načinu.



Slika 5: Povezava 4-žilnega kabla

2.2 Posredovanje podatkov na Android - Bluetooth

2.2.1 Kratka zgodovina

Začetki tehnologije Bluetooth so v letu 1994, ko so v podjetju Ericsson želeli zamenjati dotedanji način povezav RS-232 z brezžično povezavo. Na podobno idejo sta prišli tudi podjetji Intel in Nokia v tistem času. Da bi bila povezava univerzalna, so se ta podjetja povezala v skupino SIG skupaj s podjetji Toshiba in IBM. Kratka zgodovina razvoja je razvidna iz tabele 7.

Leto	Dogodek
1998	Ustanovljena skupina The Bluetooth Special Interest Group (SIG)
1999	Izdana specifikacija Bluetooth 1.0
2000	Prvi izdelki (mobilni telefon, kartica za osebni računalnik, miška …)
2002	Organizacija IEEE potrdi specifikacijo 802.15.1
2003	SIG sprejel specifikacijo Bluetooth 1.2; prodanih 1 milijon izdelkov s podporo povezave Bluetooth na teden
2004	SIG sprejel specifikacijo Bluetooth 2.0 Enhanced Data Rate (EDR); prodani 3 milijoni izdelkov s podporo povezave Bluetooth na teden
2005	Prodanih 5 milijonov izdelkov s podporo povezave Bluetooth na teden
2006	Prodaja doseže 10 milijonov izdelkov s podporo povezave Bluetooth na teden
2009	SIG sprejel specifikacijo Bluetooth 3.0; napove energijsko varčno verzijo 4.0
2011	SIG sprejel specifikacijo Bluetooth 4.0
2013	SIG sprejel specifikacijo Bluetooth 4.1

Tabela 7: Zgodovina protokola Bluetooth [9]

2.2.2 O protokolu Bluetooth

Protokol deluje na frekvenci 2,4 GHz, kar pomeni, da si deli frekvenčni prostor z drugimi tehnologijami (npr. brezžično omrežje Wi-fi, mikrovalovna pečica …) in je tako podvržen tudi motnjam s strani teh naprav [10]. Da bi se temu izognili, ima protokol krajše podatkovne paketke in lahko hitro menja frekvence znotraj svojega frekvenčnega spektra. Tako je bolj varen in odporen na motnje.

Z menjavanjem frekvence je doseženo, da pokriva celoten spekter ISM (Industrial, scientific and medical) in upošteva omejitve ISM. Ker se frekvence menjavajo redno, pomeni, da če pride do motenj in napak, se paket pošlje ponovno na naslednji frekvenci, kjer verjetno ni več motenj. Pomaga pa tudi pri varnosti, saj je težje slediti signalu, če se frekvenca redno menjava.

Protokol Bluetooth deluje po principu gospodar/suženj (master/slave). Naprava, ki je gospodar na omrežju, lahko poveže do 7 naprav sužnjev. Gospodar pošlje vsaki od naprav unikatni naslov in vrednost notranje ure. S tema dvema informacijama vsaka naprava po istem algoritmu točno izračuna, katera je naslednja frekvenca.

2.2.3 Modul Bluetooth HC-05

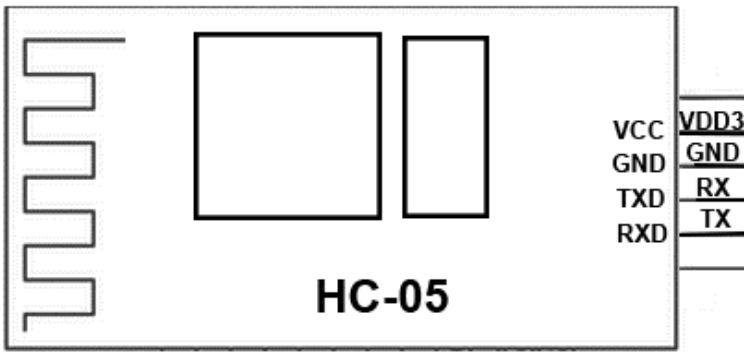
Za povezavo mikrokrmilnika ARM STM32F407 Discovery s tablico Android smo potrebovali povezavo Bluetooth. Na mikokrmilnik smo povezali modul Bluetooth HC-05, ki se z mikrokrmilnikom sporazumeva preko povezave UART.

Modul Bluetooth HC-05 ima lastnosti:

- moč oddajanj do +4dBm
- delovanje na 1,8-3,6V
- nadzor PIO
- nastavljivo hitrost UART
- nastavljanje lastnosti preko ukazov AT

Povezavo smo izvedli preko 4 pinov, kot je razvidno na sliki 6:

- UART Tx – pošiljanje
- UART Rx – prejemanje
- ozemljitev
- napajanje 3V



Slika 6: Modul Bluetooth HC-05

2.3 Android - prikazovanje informacij

2.3.1 Sistem Android

Prvotno je bil operacijski sistem, ki ga je razvijalo podjetje Android, namenjeno digitalnim fotoaparatom. Ker pa je bil trg premajhen, so se preusmerili na napreden operacijski sistem za mobilne telefone. Podjetje je začelo z razvojem leta 2003, že leta 2005 pa ga je kupilo podjetje Google. Pod okriljem novega podjetja je ekipa starega podjetja razvila sistem na podlagi jedra Linux. Skupina proizvajalcev je v povezavi z Googlom leta 2007 ustanovila konzorcij Open Handset Alliance s ciljem razviti odprti standard za mobilne telefone [11].

Prvi komercialni telefon z operacijskim sistemom Android je bil HTC Dream, ki je bil dan na trg oktobra 2008. V naslednjih letih je operacijski sistem Android doživel hiter razvoj in v letu 2014 obsega skoraj 85% trga mobilnih telefonov [12].

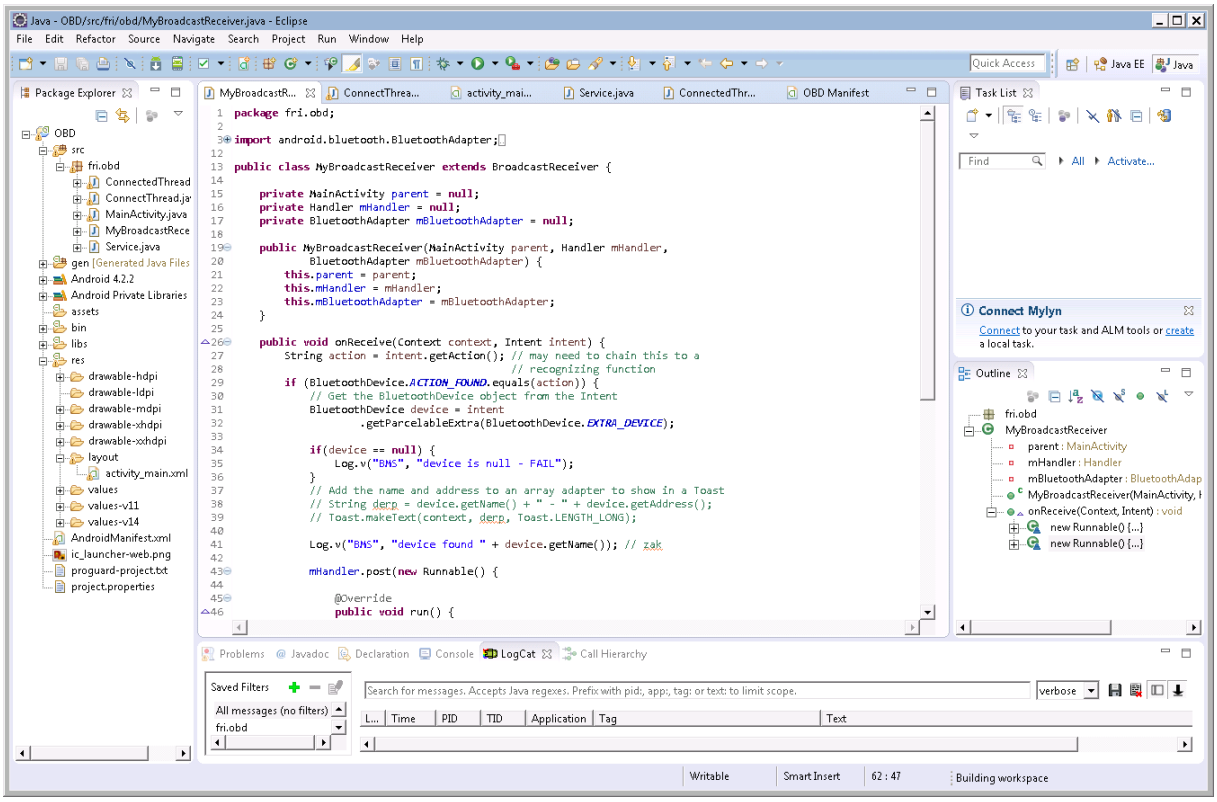
2.3.2 Naprava Android

Android se danes uporablja za različne elektronske naprave, predvsem pa za elektronske tablice in mobilne telefone. Za naš sistem smo uporabili tablico Lenovo A3000 H, na katero je bil nameščen Android 4.2.2. Jelly Bean 4.2 je na sliki 7.

Slika 7: Tablica Lenovo A3000 H

2.3.3 Okolje Eclipse in dodatek ADT

Aplikacije za operacijski sistem Android se po navadi razvijajo z uporabo razvojnega paketa Android Software Development Kit. Uporabljajo pa se lahko tudi druga orodja, v našem primeru razvojno okolje Eclipse z dodatkom ADT (Android Development Kit).



Slika 8: Okno razvojnega okolja Eclipse z ADT

Na sliki 8 vidimo tipično razporeditev razvojnega okolja:

- na levi strani drevesna struktura projekta s pomembnimi datotekami
- osrednji del, kjer se programira (zavihki s kodo)
- desni del s kratkimi povzetki kode
- spodnji del z dnevnikom

Uporabnost okolja z ADT je tudi, da ni vedno potrebna naprava za razvoj, ampak se lahko koda zaganja na simulatorju, ki ga vsebuje. Žal ga v našem primeru nismo mogli uporabiti, saj simulator ne simulira prenosa Bluetootha.

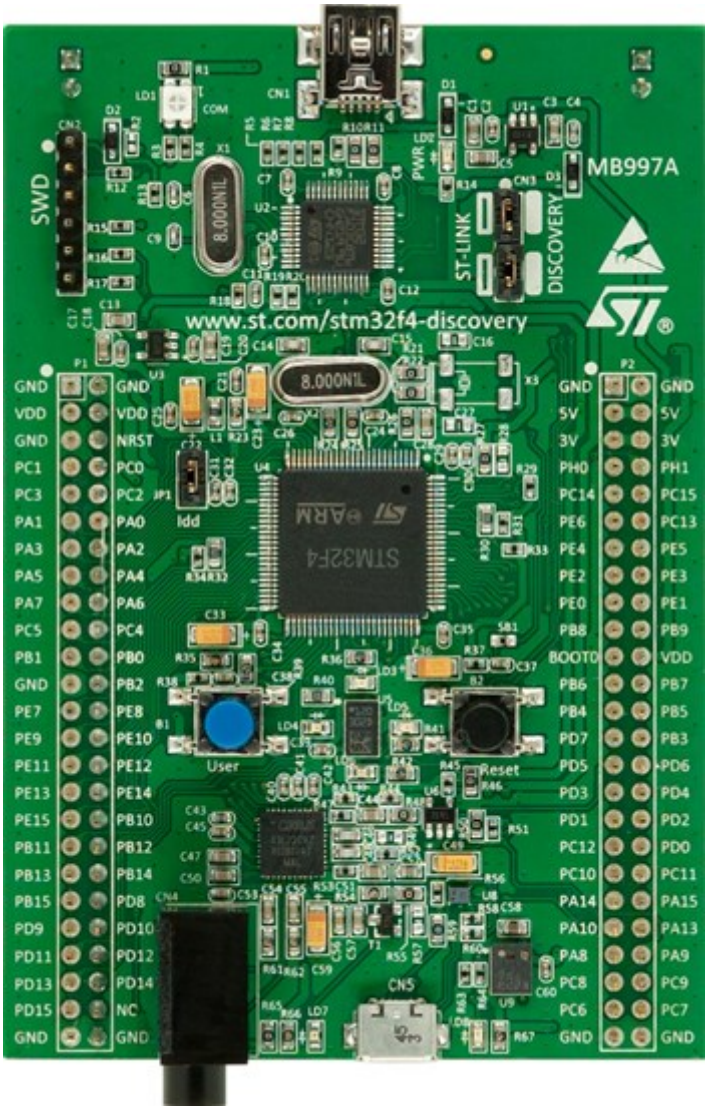
2.4 ARM STM32F407 Discovery - osnovni sistem

2.4.1 Lastnosti razvojne ploščice

Za osnovo našega diagnostičnega sistema smo uporabili razvojno ploščico ARM STM32F407 Discovery s slike 9 podjetja STMicroelectronics. Osnovan je na arhitekturi ARM Cortex M4, ki je zelo zmogljiva. Nekaj lastnosti razvojne ploščice [13]:

- 1 MB Flash
- 192 KB RAM
- pakiranje LQFP100
- priklop ST-LINK/V2 priklop (SWD priklop)
- 5V napajanje preko napajalnika ali USB
- napajanje 3V ali 5V
- 3-osni pospeškomer LIS302DL/LIS3DSH ST MEMS
- MP45DT02, zvočno tipalo ST MEMS, večsmerni digitalni mikrofon
- CS43L22 zvokovni DAC
- 8 LED diod
- 2 gumba
- USB OTG z mikro USB priklopom

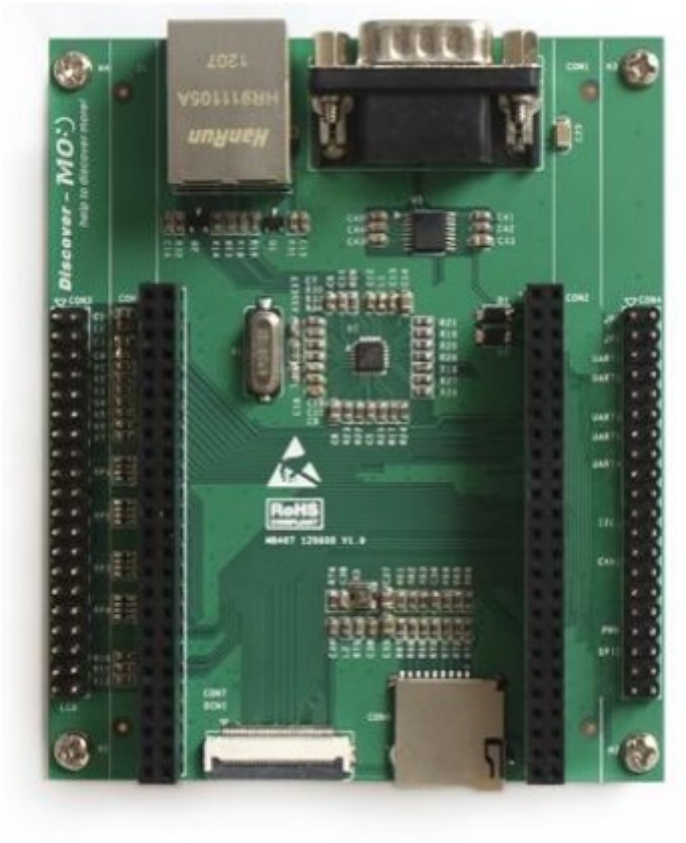
Uporabljena razvojna ploščica je zelo primerna za začetnike, saj je na voljo širok nabor že pripravljenih primerov in se lahko hitro razvija aplikacije zanjo.



Slika 9: Razvojna ploščica ST32F407 Discovery [14]

2.4.2 Razširitvena ploščica »BaseBoard«

Razširitvena ploščica »Base Board STM32F4DIS-BB« je razširitveni dodatek, ki zelo olajša delo z uporabljeno razvojno ploščico. Na sami plošči je že integrirana podpora za kartice SD, omrežni priključek 10/100, priklop za kamero in priklop za LCD (Slika 10). Na voljo je tudi možnost priklopa za UART, I2C, SPI, CAN, PWM in GPIO. V našem primeru je bil uporabljen za testiranje povezave UART najprej z osebnim računalnikom ter na koncu tudi priklop UART z modulom Bluetooth.



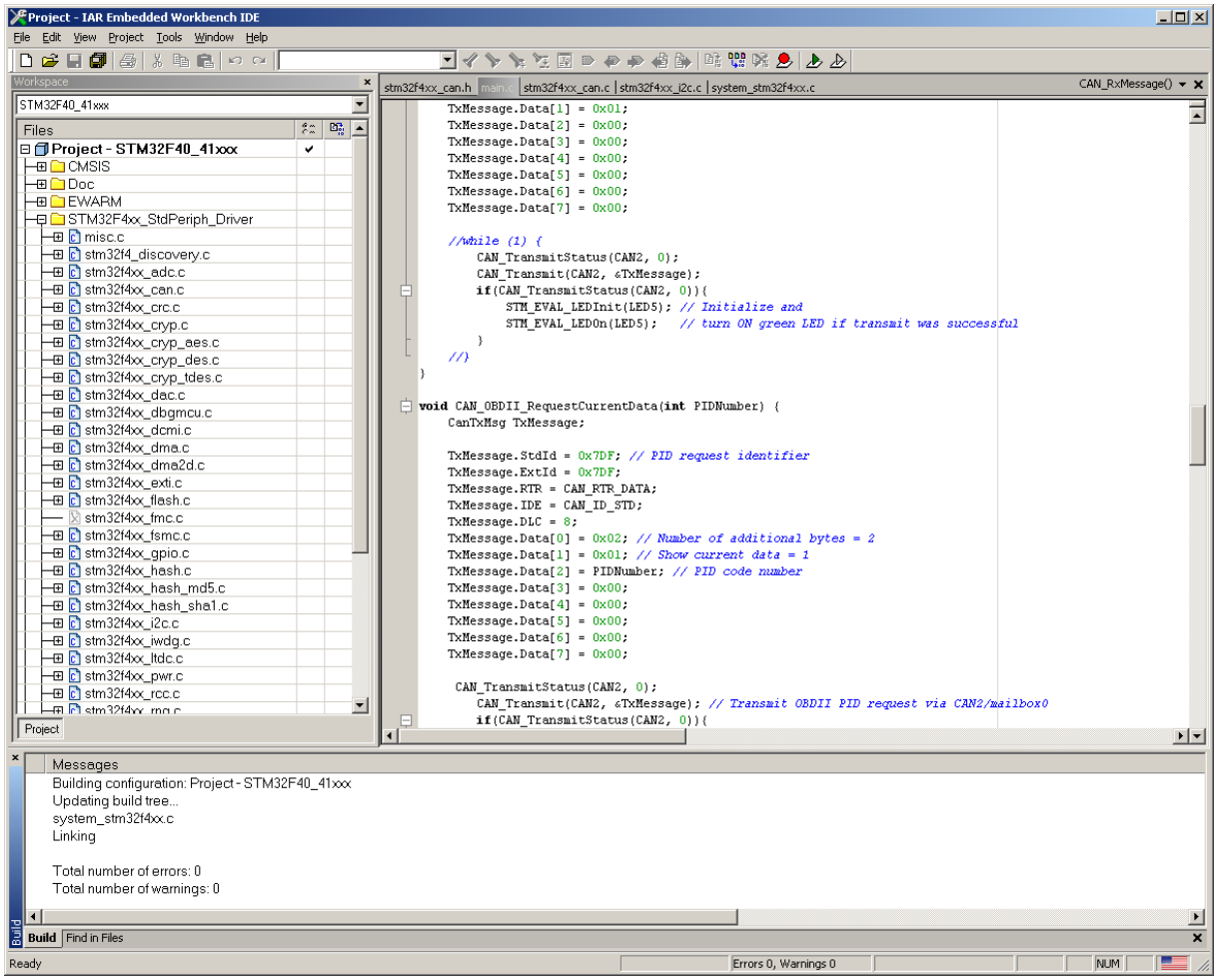
Slika 10: Dodatek STM32F4DIS-BB [15]

2.4.3 EWARM okolje

Razvojno okolje EWARM (»IAR Embedded Workbench for ARM«) je namenjeno razvoju aplikacij za mikrokrmilnike podjetja STMicroelectronics. Obstajajo tudi alternativna razvojna okolja, kot je CooCox, a so manj pogosto uporabljena.

Osnovna razporeditev oken na vmesniku razvojnega okolja EWARM (Slika 11):

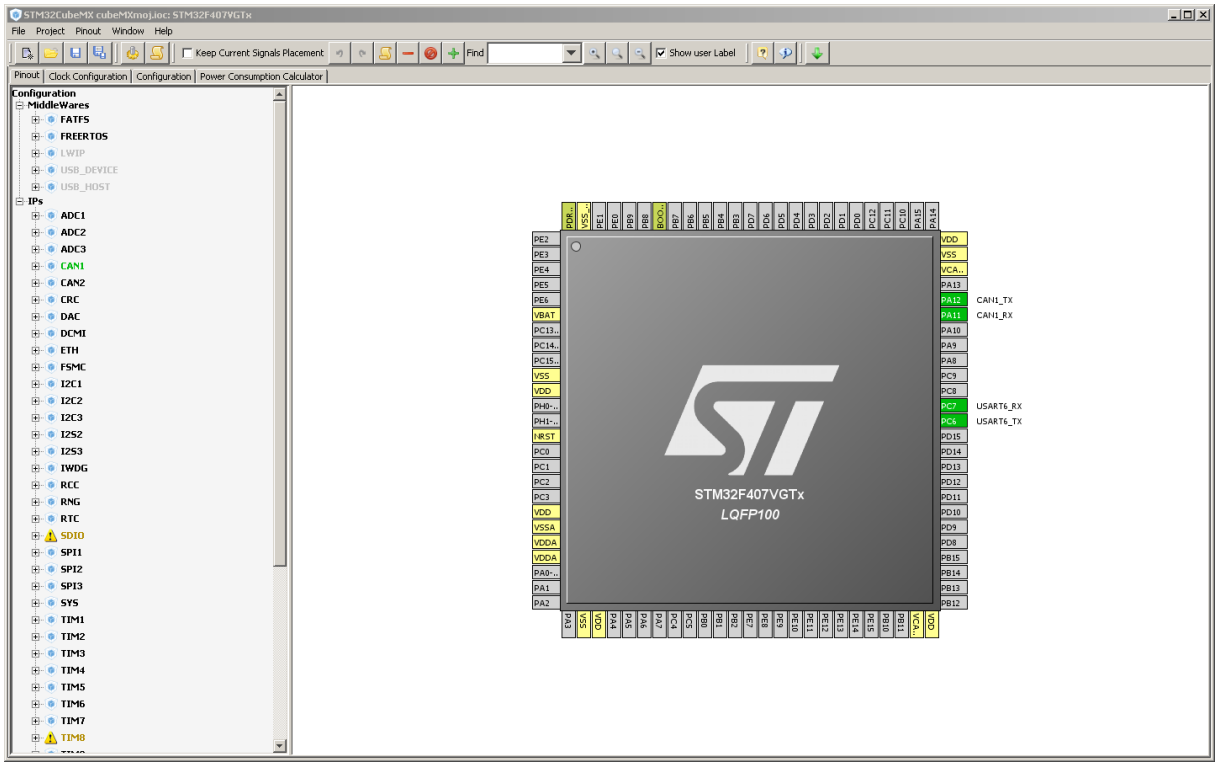
- na levi strani je okno z osnovno strukturo projekta in s knjižnicami
- desno je glavno okno, kjer se programira
- spodaj je izhod ob prevajanju in analizi programa



Slika 11: Razvojno okolje EWARM

2.4.4 Program cubeMX

Program cubeMX (Slika 12) je novejša pridobitev razvijalcev za lažji začetek razvoja aplikacij. Omogoča, da na podlagi čarovnika z izbiro ustreznih lastnosti uporabljenega mikrokrmilnika hitro določimo konfiguracijo projekta – izberemo, kaj želimo uporabiti na mikrokrmilniku, in čarovnik nam določi pine ter opozori na morebitne konflikte. Ob izdelavi projekta tudi doda k projektu potrebne knjižnice in pripravi začetne nastavitve priključkov.



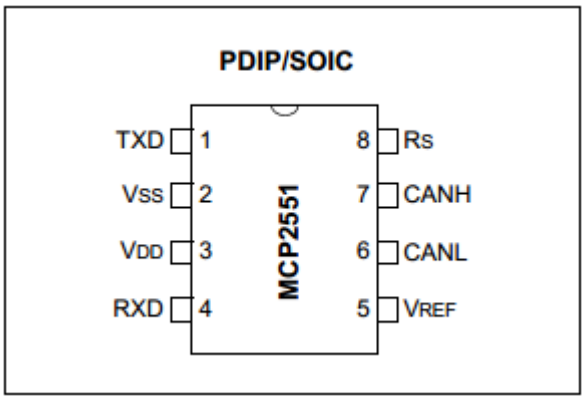
Slika 12: Program cubeMX

Poglavje 3 Implementacija z ARM STM32F407

3.1 Povezava CAN in STM32F407

3.1.1 Vezava

Modul CAN smo povezali z razvojno ploščico STM32F407 Discovery, saj potrebujemo povezavo z avtomobilsko diagnostiko OBD II, ki deluje na protokolu CAN. Razvojna ploščica sicer že podpira 2 vmesnika CAN, nima pa sprejemno-oddajnega modula, da bi povezava delovala. To rešuje modul CAN, ki ima vgrajen sprejemno-oddajni čip CAN MCP2551 (Slika 16).

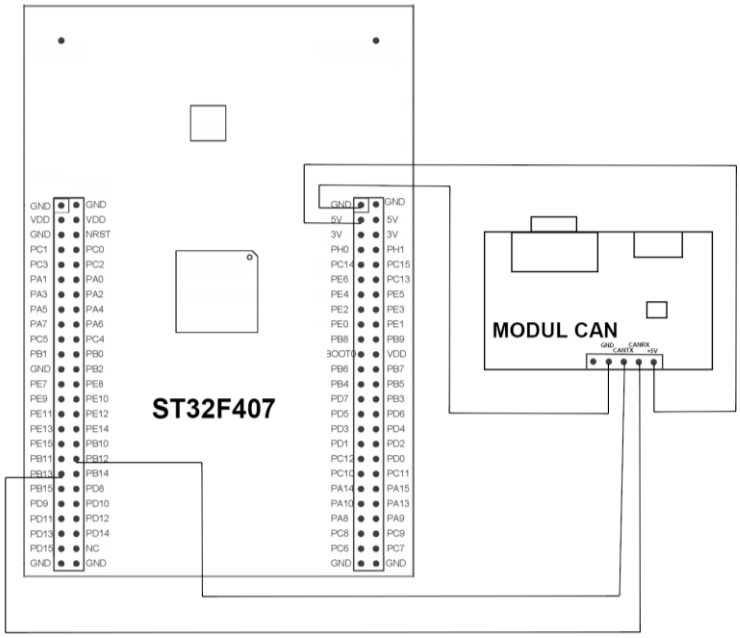


Slika 13: MCP2551 [16]

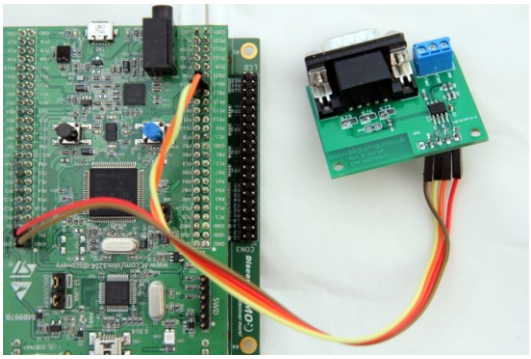
Sprejemno-oddajni čip CAN prejme z razvojne ploščice signala Tx in Rx ter napajanje, na omrežje CAN pa pošilja signala CAN-H (High) in CAN-L (Low) (Slika 14 in slika 15). Na strani razvojne ploščice pa smo priklopili pine kot kaže tabela 8.

modul CAN	STM32F407
napajanje 5V	napajanje 5V
CAN_RX	PB13 (TX
CAN_TX	PB12 (RX)
ozemljitev	ozemljitev

Tabela 8: Povezava CAN s ST32F407



Slika 14: Shema povezave modula CAN in razvojne ploščice



Slika 15: Vezava modula CAN in razvojne ploščice

3.1.2 Programski del

Najprej vklopimo uro na vodilu AHB1 ter uro za CAN1 in CAN2. Pravilo je, da se vklopi ura za oba priključka CAN, čeprav potrebujemo le za CAN2, ki deluje v načinu suženj (»slave«) [8]. Avtomobilski del je v načinu gospodar (»master«).

```
/* ura */

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN2, ENABLE);
```

Nato moramo ustrezno nastaviti lastnosti pinov za sprejem in oddajanje na mikrokrmilniku.

```
/* primer za nastavitev pina za pošiljanje */

GPIO_InitStructureCAN_TX.GPIO_Pin = GPIO_Pin_13;
GPIO_InitStructureCAN_TX.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructureCAN_TX.GPIO_OType = GPIO_OType_PP
GPIO_InitStructureCAN_TX.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructureCAN_TX.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_PinAFConfig(GPIOB, GPIO_PinSource13, GPIO_AF_CAN2);
```

Sledi nastavitev povezave CAN, ki bo delovala preko pinov.

```
/*  nastavitev povezave */

 CAN_InitStructure.CAN_TTCM = DISABLE;
CAN_InitStructure.CAN_ABOM = DISABLE;
CAN_InitStructure.CAN_AWUM = DISABLE;
CAN_InitStructure.CAN_NART = DISABLE;
CAN_InitStructure.CAN_RFLM = DISABLE;
CAN_InitStructure.CAN_TXFP = DISABLE;
CAN_InitStructure.CAN_Mode = CAN_Mode_Normal;
CAN_InitStructure.CAN_SJW = CAN_SJW_1tq;
CAN_InitStructure.CAN_BS1 = CAN_BS1_14tq;
CAN_InitStructure.CAN_BS2 = CAN_BS2_6tq;
CAN_InitStructure.CAN_Prescaler = 4;
```

Protokol CAN deluje na podlagi filtriranja, zato moramo nastaviti filter za naše potrebe.

```
/* filter */

CAN_FilterInitTypeDef CAN_FilterInitStructure;
CAN_FilterInitStructure.CAN_FilterNumber = 27;
CAN_FilterInitStructure.CAN_FilterFIFOAssignment = CAN_FIFO0;
CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_32bit;
CAN_FilterInitStructure.CAN_FilterIdHigh = 0x0700 << 5;
CAN_FilterInitStructure.CAN_FilterIdLow = 0x0000;
CAN_FilterInitStructure.CAN_FilterMaskIdHigh = 0x0700 << 5;
CAN_FilterInitStructure.CAN_FilterMaskIdLow = 0x0000;
CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
```

Zahtevo za podatke pa pripravimo odvisno od tega, kaj želimo prejeti, in jo pošljemo. Nastavimo, na katero enoto želimo poslati zahtevo, način delovanja in parameter PID za podatek, ki ga želimo.

```
/* zahteva */

TxMessage.StdId = 0x7E0;
TxMessage.ExtId = 0x7E0
TxMessage.RTR = CAN_RTR_DATA
TxMessage.IDE = CAN_ID_STD;
TxMessage.DLC = 8;
TxMessage.Data[0] = 0x02;
TxMessage.Data[1] = 0x01;
TxMessage.Data[2] = PIDNumber;
TxMessage.Data[3] = 0x00;
TxMessage.Data[4] = 0x00;
TxMessage.Data[5] = 0x00;
TxMessage.Data[6] = 0x00;
TxMessage.Data[7] = 0x00;
```

Prejmemo pa podatke, kjer je naveden način delovanja, parameter PID in vrednosti za izračun le-tega.

```
/* odgovor */

d0[0] = RxMessage.Data[0];
d0[1] = RxMessage.Data[1];
d0[2] = RxMessage.Data[2];
d0[3] = RxMessage.Data[3];
d0[4] = RxMessage.Data[4];
d0[5] = RxMessage.Data[5];
d0[6] = RxMessage.Data[6];
d0[7] = RxMessage.Data[7];
```

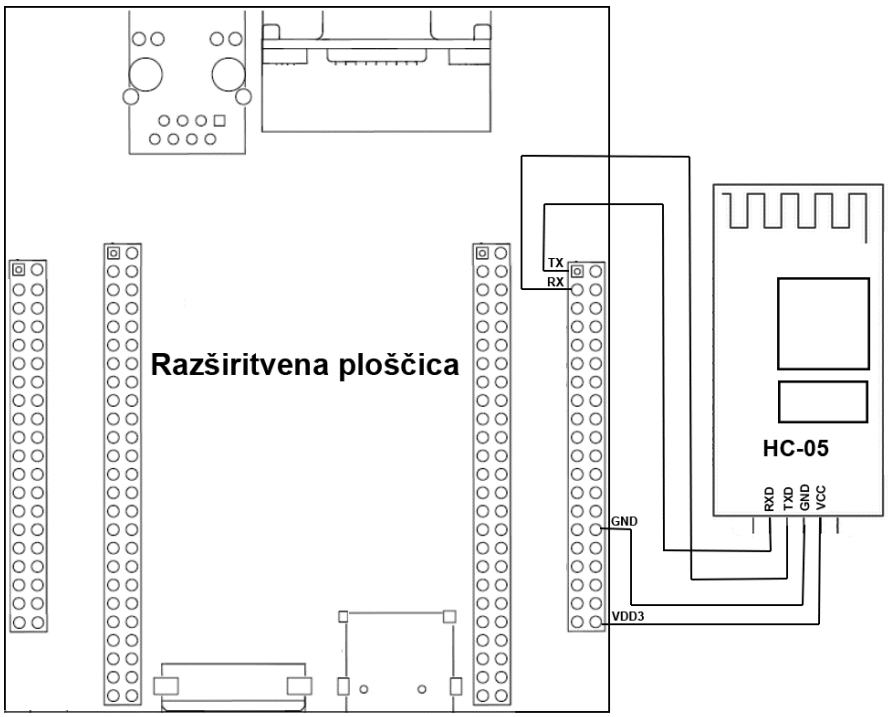
3.2 Povezava Bluetooth in STM32F407

3.2.1 Vezava

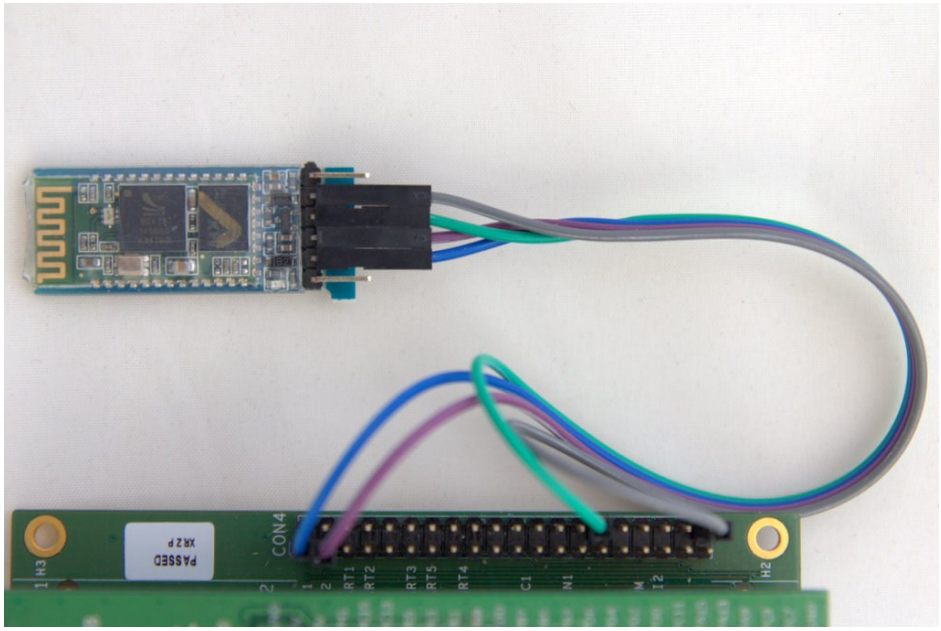
Modul Bluetooth HC-05 smo potrebovali za povezavo mikrokrmilnika z napravo Android. Modul se povezuje preko povezave UART z mikrokrmilnika. Uporabili smo UART6, saj smo ga lahko predhodno dobro testirali s pomočjo razširitvene ploščice.. Razširitvena ploščica ima namreč že priključek DB9 in čip MAX232, ki omogoča neposreden priklop na serijski priključek osebnega računalnika. Tako smo lahko preko terminalskega programa na osebnem računalniku spremljali promet, ki ga pošlje mikrokrmilnik. Vezava modula Bluetooth in razširitvene ploščice je razvidna iz tabele 9, slike 16 in slike 17.

Modul Bluetooth	STM32F407
napajanje 3,3 V	napajanje 3V
ozemljitev	Ozemljitev
TXD	RXD (PC6)
RXD	TXD (PC7)

Tabela 9: Povezava modula Bluetooth s ST32F407



Slika 16: Shema vezave modula Bluetooth in razširitvene ploščice



Slika 17: Povezava modula Bluetooth z razširitveno ploščico

3.2.2 Programski del

Najprej vklopimo uro na vodilu AHB1 ter uro za USART6.

```
/* ura */

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART6, ENABLE);
```

Nato moramo ustrezno nastaviti lastnosti pinov za sprejem in oddajanje na mikrokrmilniku.

```
/* primer za nastavitev pina za pošiljanje in prejamanje */

GPIO_InitStructureUSART.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
GPIO_InitStructureUSART.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructureUSART.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructureUSART.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructureUSART.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &GPIO_InitStructureUSART);
GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_USART6);
GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_USART6);
```

Sledi nastavitev povezave UART, ki bo delovala preko pinov.

```
/*  nastavitev povezave */

USART_InitStructure.USART_BaudRate = 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
```

Nato pošljemo in prejemamo podatke preko povezave UART6, ki smo jo ustrezno nastavili.

```
/*  Koda za pošiljanje */

for (int i=0; i< meja_vhod; i++) {
while(USART_GetFlagStatus(USART6,USART_FLAG_RXNE) == RESET);
vhod[i] = USART_ReceiveData(USART6);
}


/*  Koda za prejemanje */

for (int j=0; j< meja_izhod; j++){
while(USART_GetFlagStatus(USART6, USART_FLAG_TXE) == RESET);
USART_SendData(USART6, izhod[j]);
}
```

3.3 Aplikacija Android

Aplikacija fri.OBD Android je aplikacija (Slika 18), spisana v razvojnem okolju Eclipse z dodatkom ADT. Program ob zagonu preveri povezavo Bluetooth in se poveže na povezavo, če ni povezano. V primeru, da se povezava prekine, jo lahko ponovno vzpostavimo z gumbom Poveži (Slika 19). S klikom na gumb Konec se aplikacija zaustavi in ob izhodu počisti za sabo spomin in povezave. Program prikazuje tri diagnostične parametre avtomobila:

- hitrost
- vrtljaji motorja
- temperatura hladilne tekočine

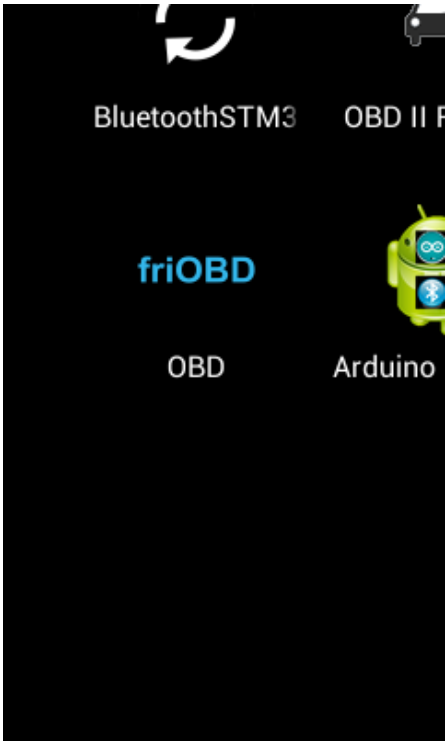
Po zagonu se program poveže s povezavo Bluetooth in pošlje 8 bajtov podatkov preko povezave mikrokrmilniku. Mikrokrmilnik razbere podatke iz zadnjih treh bajtov, ki določajo, da želi podatke hitrosti, vrtljajev in temperature hladilne tekočine. Podatek zahteva preko povezave CAN od avtomobila in nato pošlje nazaj aplikaciji 4 bajte podatkov. Na podlagi teh podatkov se izračuna hitrost, vrtljaji motorja in temperatura hladilne tekočine in prikaže na zaslonu. Podatki se obnavljajo vsako sekundo.

Aplikacija fri.OBD je bila izdelana na osnovni obstoječe aplikacije za prikaz podatkov preko povezave Bluetooth avtorja Matevža Peska.

Primer kode za pošiljanje in prejemanje preko podatkovne povezave Bluetooth. Pripravimo niz 8 znakov v šestnajstiškem zapisu, saj se pošilja v paketih po 8 bajtov. Prvih 5 je koda za številko 0 (preverjamo ob prejemu za varnost), preostale tri pa so kode za parametre PID, ki jih zahtevamo za podatke.

```
/* Pošiljanje */

private void sendInit() {
   byte[] init = hexStringToByteArray("31313131310D0C05");
   try {
            if(mmOutStream != null) {
            mmOutStream.write(init);
            mmOutStream.flush();
   }        ...
```



Slika 18: Ikona programa

Slika 19: Vmesnik programa

Ob prejemanju se prejme niz podatkov, preveri, obdela in preračuna vrednosti ter prikaže na zaslonu.

```
/* Prejemanje */

boolean parsedMessage = true;
   while (keepRunning) {
            try {
            bytes = mmInStream.read(buffer);
            count += bytes;
            if(count > 4){
                     count = 4;
            }
            System.arraycopy(buffer, 0, message, count - bytes, bytes);
            if (count == 4) {
            parsedMessage = localParseMessage();
            count = 0;
            mHandler.post(new Runnable() {
            @Override
            public void run() {
                     parent.updateGUI();
            }
   } ...
```

Primer kode vmesnika, ki je v posebni datoteki xml.

```
/* Oblika */

<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android
xmlns:tools=http://schemas.android.com/tools
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="${relativePackage}.${activityClass}" >

<TextView
android:id="@+id/hitrost_txt"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"

android:text="Hitrost"
android:textAppearance="?android:attr/textAppearanceLarge" />
   ...
```

Poglavje 4 Delovanje

Naš sistem je sestavljen iz okvirno petih večjih sklopov, ki delujejo na sledeč način:

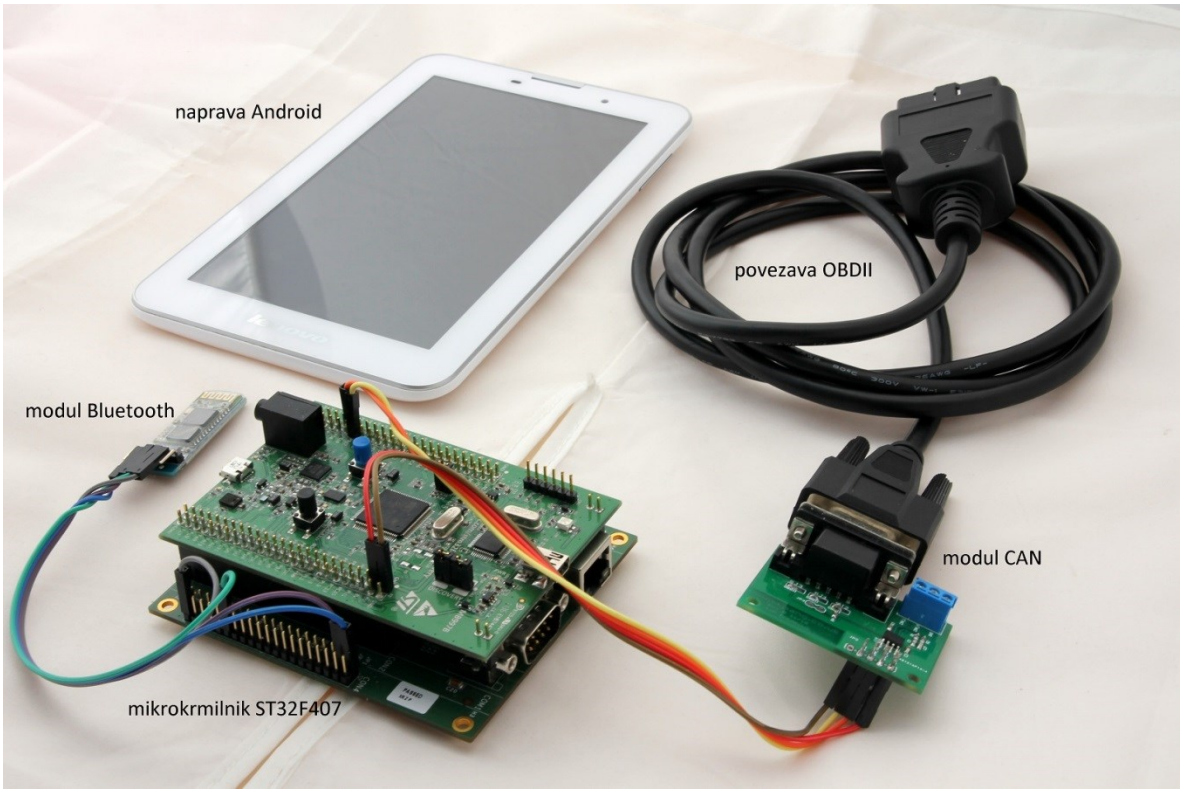
- Aplikacija fri.OBD na tablici Android, ki preko povezave Bluetooth vsako sekundo pošilja niz s podatki, kjer je določeno, katere parametre avtomobila želimo. Pošljemo niz oblike »31313131310D0C05«.

- Podatke nato sprejme modul Bluetooth HC-05, ki podatke preko povezave UART posreduje mikrokrmilniku ST32F407.

- Mikrokrmilnik tako pridobi podatke, torej niz »31313131310D0C05«. Niz je stavljen iz 16 znakov, ki v parih pomenijo šestnajstiško predstavitev nekega znaka. Tako »31« pomeni znak »0«. Mikrokrmilnik najprej preveri, če je vodilnih pet znakov enako »0«. Če to drži, pomeni, da je prejel ustrezno zahtevo za podatke. Če bi bil v prvih petih parih kak drug podatek, bi pomenilo, da je prišlo do napake v prenosu, in bi podatke zavrgli. V primeru prenosa brez napake se zabeležijo zadnji trije pari, ki pomenijo: »0D« koda za hitrost, »0C« koda za vrtljaje in »05« koda za temperaturo hladilne tekočine.

- Modul CAN pošlje tri zahteve preko kabla OBDII avtomobilu na priključek OBDII. Za vsak podatek se pošlje ena zahteva.

Za lažjo predstavo so označeni različni sklopi sistema na sliki 20. Pot podatkov v nasprotno smer gre na isti način. Pošljejo se štirje podatki. Ko jih prejme aplikacija fri.OBD, izračuna hitrost, vrtljaje in temperaturo. Na koncu se podatki osvežijo na vmesniku aplikacije. Vmesnik ima še dva gumba. Z gumbom »Poveži« ponovno vzpostavimo povezavo Bluetooth v primeru, da se prekine in se podatki ne osvežujejo. Gumb »Konec« pa prekine izvajanje programa, prekine povezavo Bluetooth ter počisti povezave.



Slika 20: Različni sklopi sistema

Poglavje 5 Sklepne ugotovitve

Med izdelavo sistema smo se srečali z nekaterimi težavami. Ker razvojna ploščica ST32F407 Discovery deluje s frekvenco 168MHz, smo morali v datoteki stm32f4xx.h nastaviti frekvenco kristala (HSE_value), ki ga uporablja razvojna ploščica (8MHz). V datoteki system_stm32f4xx.c pa smo morali nastaviti delitelj (PLL_M), ki je v primeru naše razvojne ploščice določen na vrednost 8. Žal začetnik, ki se šele spoznava s krmilnikom, tega ne ugotovi takoj in marsikateri vnaprej pripravljen primer iz razvojnega okolja ne deluje takoj. Dokler tega ustrezno nismo nastavili, ni delovala povezava UART.

Težavo smo imeli tudi s tehnično nezanesljivimi komponentami. Poceni modul Bluetooth HC-05 prihaja na trg v različnih verzijah, razporeditvah pinov in različnih programskih verzijah. Uporaba je tako otežena, pa tudi povezava Bluetooth ni bila vedno stabilna. Dodati je bilo potrebno tudi kak časovni zamik, da je povezava delovala bolje. Ker ima modul dodatne zaščite pred preobremenitvijo, je potrebno programsko nastaviti sprejemni pin na način »PullUp«, drugače modul ne more sprejemati podatkov. Žal pa tega dokumentacija ne opiše, ampak je bilo potrebno ugotoviti s prebiranjem forumov in izkušenj uporabnikov.

Sistem je bil zasnovan tako, da uporabi različne načine komunikacije in področja, ki jih v času študija nismo spoznali prav podrobno. Uporabi se povezava CAN, povezavi Bluetooth in UART. Za osnovo je mikrokrmilnik ST32F407, s katerim smo se prvič srečali. Podobno z napravo Android in programiranjem v programskem jeziku Java in okolju Eclipse.

Sistemov za diagnostiko OBDI je mnogo, a so narejeni na drugih pristopih, ne na mikrokrmilniku ST32F407. Sistem je glede na druge okoren in prevelik, a je dobro izhodišče za učenje novih tehnologij. Na podlagi pridobljenih izkušenj se lahko izvede marsikateri drug izdelek. Imamo žično in brezžično povezavo, tekom testiranja smo morali povezati mikrokrmilnik tudi z računalnikom, kar je prav tako uporabno. Povezava z Androidom nam tudi nudi mnoge možnosti povezovanja s spletom in hranjenja podatkov.

Sistem bi lahko nadgradili tako, da bi naprava Android bila kot osnovna informacijska naprava v avtomobilu namesto sedanjih avdio sistemov, povezana pa bi bila tudi na splet preko mobilnega omrežja in lokacijsko osveščena preko signala GPS. Pridobili bi lahko inteligentni sistem, ki podaja informacije vozniku, sporoča pa njegovo lokacijo in parametre avtomobila na splet ali zunanjemu opazovalcu.

Literatura

[1] „Controller Area Network (CAN) Overview,“ National Instruments, 2014. [Elektronski]. Dostopno: http://www.ni.com/white-paper/2732/en/.

[2] „CAN in Automation (CiA): Controller Area Network (CAN),“ CIA, 2014. [Elektronski]. Dostopno: http://www.can-cia.org/index.php?id=systemdesign-can.

[3] „CAN protocol specification,“ CAN-CIA, 2014. [Elektronski]. Dostopno: http://www.can-cia.org/index.php?id=164.

[4] „Wikipedia,“ 2014. [Elektronski]. Dostopno: http://en.wikipedia.org/wiki/CAN_bus.

[5] „CAN physical layer,“ CAN CIA, 2014. [Elektronski]. Dostopno: http://www.can-cia.org/index.php?id=systemdesign-can-physicallayer.

[6] „Wikipedia - File:OBD connector shape.svg,“ 2014. [Elektronski]. Dostopno: http://en.wikipedia.org/wiki/File:OBD_connector_shape.svg.

[7] „Wikpedia - OBD-II PIDs,“ 2014. [Elektronski]. Dostopno: http://en.wikipedia.org/wiki/OBD-II_PIDs.

[8] J. Cernatič, Sistem za avtomobilsko diagnostiko 39-41, Ljubljana, 2012.

[9] „History of the Bluetooth Special Interest Group,“ 2014. [Elektronski]. Dostopno: http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx.

[10] „Bluetooth Technology Basics,“ 2014. [Elektronski]. Dostopno: https://developer.apple.com/library/mac/documentation/devicedrivers/conceptual/bluetooth/BT_Bluetooth_Basics/BT_Bluetooth_Basics.html..

[11] „Android (operating system),“ 2014. [Elektronski]. Dostopno: http://en.wikipedia.org/wiki/Android_(operating_system).

[12] „IDC: Smartphone OS Market Share,“ 2014. [Elektronski]. Dostopno: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[13] „Discovery kit for STM32F407/417 lines - with STM32F407VG MCU,“ 2014. [Elektronski]. Dostopno: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00037955.pdf.

[14] „Discovery kit for STM32F407/417,“ STMicroelectronics, 2014. [Elektronski]. Dostopno: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419.

[15] „http://www.element14.com/,“ 2014. [Elektronski]. Dostopno: http://www.element14.com/community/servlet/JiveServlet/downloadBody/51090-102-2-267814/STM32F4DIS-BB%20for%20the%20Base%20Board.pdf.

[16] „Microchip,“ 2014. [Elektronski]. Dostopno: http://ww1.microchip.com/downloads/en/DeviceDoc/21667f.pdf.